

Portable Program Review

Vol. 1, No. 5

November, 1985

FAVORITE SOAP OPERA

SOAP.BA, by Richard S. Elliot, M.D., isn't a computerized gossip tabulator for General Hospital. SOAP is an acronym:

S - Subjective
O - Objective
A - Assessment
P - Plan

The SOAP program is geared to anyone whose function is to resolve complaints. "Subjective" indicates the problem expressed by the patient or client. "Objective" denotes the evaluation of the problem by an expert. "Assessment" covers the tentative diagnosis or appraisal by the problem-solver. "Plan" includes the final diagnosis, methods for resolving the problem and all costs.

It Doesn't Work

The family car doesn't run right.

Subjective: The owner says to the mechanic, "The car runs poorly and gets low gas mileage."

Objective: The mechanic notices that the choke doesn't work and the oil hasn't been changed in 10,000 miles.

Assessment: The car needs a tune-up and oil change, along with a road test.

Plan: Perform the tune-up, replacing worn parts. Also, impress on the owner the need for regular, routine maintenance.

Soapy Waters

SOAP.BA isn't the world's most sophisticated program -- but it uses the principles above to organize the process of resolving problems.

The BASIC program simply prompts for one- or two-line descriptions of each of the four SOAP stages. The answers, along with a one-line identifier for the problem and the date, are printed or saved to RAM.

Why is this process useful? Documentation. If the problem-solver, like the auto mechanic, could document the entire car-repair transaction, everyone would be kept happy. The service station's records would be up-to-date, the customer would know that his concern was recorded, and the mechanic would have written notes in case of future complaints.

Another application: Artificial Intelligence (AI). Certainly,

IN THIS ISSUE:

FAVORITE SOAP OPERA1

CHECKING
THE CHECKLIST2

ALL SORTS OF SORTS.....4

A MATTER OF TIMING5

SOAP.BA isn't an AI program. But artificial intelligence attempts to reason -- and learn. SOAP-type records represent pre-learned experience in an organized format. So, if many years' SOAP records are stored in an expert system (an AI problem-solving computer system), and a customer reports that his car runs poorly and gets low gas mileage, the computer would have a good chance at diagnosis and cost estimation.

Listing of SOAP.BA, the problem recorder.

```

100 REM S.O.A.P. Program
110 REM
120 DEFSTR A-I
130 CLEAR 25000,MAXRAM
140 CLS
150 PRINT @ 86, "Custom Medical
    Software-1000"
160 PRINT @ 132, "42 Walnut Lane"
170 PRINT @ 172, "Davis, CA 95616"
180 PRINT @ 252, "Copyright 1984"
190 PRINT @ 290, "All Rights
    Reserved"
200 FOR N=1 TO 1500
210 IF INKEY$ <> "" THEN 230
220 NEXT N
230 CLS
240 PRINT "Please enter the file
    name."
250 PRINT "Note that LPT: may be
    used for the line printer:"
260 LINE INPUT NP$
270 IF INSTR(NP$,".") = 0 THEN NP$
    = NP$ + ".DO"
280 OPEN NP$ FOR OUTPUT AS 1
290 CLS
300 LINE INPUT "Name and Problem:";
    A
310 CLS
320 LINE INPUT "Subjective:";B
330 IF LEN(B) => 249 THEN LINE
    INPUT "Subj. Cont'd.:";C
340 CLS
350 LINE INPUT "Objective: ";D
360 IF LEN(D) => 249 THEN LINE
    INPUT "Obj. Cont'd.:";E
370 CLS
380 LINE INPUT "Assessment: ";F

```

```

390 IF LEN(F) => 249 THEN LINE
    INPUT "Assess. Cont'd.:";G
400 CLS
410 LINE INPUT "Plan: ";H
420 IF LEN(H) => 249 THEN LINE
    INPUT "Plan Cont'd.:";I
430 PRINT #1, DATE$; ": " ;A
440 PRINT #1,
450 PRINT #1, "S: ";B;C
460 PRINT #1,
470 PRINT #1, "O: ";D;E
480 PRINT #1,
490 PRINT #1, "A: ";F;G
500 PRINT #1,
510 PRINT #1, "P: ";H;I
520 PRINT #1,
530 CLOSE 1
540 END

```

CHECKING THE CHECKLIST

The plane's leaving in a couple of hours: Do you have everything packed? Important presentation today: Are all the materials in the conference room? The newsletter deadline is tomorrow: Have all the stories been edited?

Ancient Man had a method of dealing with situations like the above -- the checklist. It's easy. Write down all the items, and as they're ready place a check mark next to each. Then, when every item is checked off, everything is set.

That definition of the checklist almost reads like a computer software algorithm. So, the obvious next step is to write a checklist program for the Tandy portable computers, right?

Nathaniel F. Ireland of Marlow, New Hampshire is the author of LISTXR.BA for the Model 100 and Tandy 200. This short BASIC program scans through ASCII data files line by line, offering the operator the chance to check or skip each item. When the entire list has been read,

a printable file is created that indicates the status of the checklist.

The input text file, which can have any allowable name, should have as the first line the name of the checklist. Beginning in column one, each following line should contain one checklist item. It's best to keep each checklist item as short as can be meaningful.

The LISTXR.BA program first prompts for the data file name. Checklist items are then displayed, seven at a time, on the Model 100 or Tandy 200 screen. A "Goodyear Blimp"-type scrolling message explains the available commands: "Press <X> to check off an item, <ENTER> to skip an item. <ESC> to end."

When every item has been either checked or skipped, the program ends and returns to the system's Main Menu. The output file, LISTIT.DO, contains a print-ready version of the checklist showing which items are okay -- and which are outstanding.

Applications for LISTXR.BA: Packing for a camping trip, indexing thoughts for an essay, organizing errands for Shopping Day. But don't try to keep grocery lists on laptop: It's too cumbersome, at least with this generation of software.

```

100 'LISTXR.BA - A CHECK LIST
    PROGRAM FOR THE MODEL 100
110 'by N. F. Ireland, July 20,
    1985
120 'RIGHT GRANTED TO USE and
    PUBLISH BUT NOT TO SELL
130 '
140 MAXFILES = 2
150 CLEAR 800
    :DEFINT N
    :DEFSTR L
    :DIM LL(100)
    :OPEN "RAM:LISTIT.DO" FOR OUTPUT
    AS 1
160 PRINT #1,"    LISTXR.BA CHECK
    LIST"

```

```

:PRINT #1,"    ";DATE$
:PRINT #1," "
170 CLS
:PRINT @120,;
:INPUT"Enter filename of list";LF
180 OPEN "RAM:" + LF + ".DO" FOR
    INPUT AS 2
:CLS
190 LZ = SPACE$(40) + "Press <X> to
    check off an item, <ENTER> to
    skip an item. <ESC> to end...
    We " + CHR$(158) + " PPR!"
200 NX = LEN(LZ)
210 '
220 ' Print the LISTIT.DO heading
    for the output file
230 '
240 LINE INPUT #2,LL
:PRINT #1," " + LL
:PRINT #1," "
250 '
260 'Load the array
270 '
280 IF EOF(2) THEN 300
290 NB = NB + 1
:LINE INPUT #2, LL(NB)
:GOTO 280
300 CLOSE 2
:NC = 1
310 '
320 'Screen print the items
330 '
340 FOR NI = 1 TO NB
:PRINT TAB(10);LL(NI)
:NT = NT + 1
:IF NT = 7 OR NI = NB THEN NT = 0
:GOTO 360
350 NEXT NI
360 PRINT @9 + NN,">";
370 '
380 'Scroll routine
390 '
400 FOR NZ = 1 TO NX
:PRINT @280,MID$(LZ,NZ,38)
+ " ";
410 FOR NY = 0 TO 5
:LA = INKEY$
:IF LA <> "" THEN 460
420 NEXT NY
:NEXT NZ
:GOTO 400
430 '

```



```

440 'Process selection and screen
    control
450 '
460 IF LA = "X" AND NN < 201 AND NC
    < NB OR LA = "x" AND NN < 201
    AND NC < NB THEN PRINT @9 +
    NN,"X";
    :NN = NN + 40
    :NC = NC + 1
    :PRINT @9 + NN,">";
    :GOTO 420
470 IF LA = "X" AND NC < NB OR LA =
    "x" AND NC < NB THEN CLS
    :NN = 0
    :NC = NC + 1
    :GOTO 350
480 IF LA = "X" AND NC = NB OR LA =
    "x" AND NC = NB THEN PRINT #1,
    " "
    :LA = CHR$(27)
490 IF LA = CHR$(27) THEN CLOSE
    :MAXFILES = 0
    :MENU
500 IF LA = CHR$(13) THEN 510
    ELSE 520
510 PRINT #1,SPACES(5);LL(NC)
    :IF NN < 201 AND NC < NB THEN
    PRINT @9 + NN," "
    :NN = NN + 40
    :NC = NC + 1
    :PRINT @9 + NN,">";
    :GOTO 420 ELSE LA = "X"
    :GOTO 470
520 GOTO 420

```

ALL SORTS OF SORTS

These sorting subroutines were written by Namir Clement Shammass, Professional Computing contributing editor. Readers are encouraged to submit their own special-purpose subroutines. In addition to the BASIC listing, each submission should have the following parts:

- * A clear statement of the purpose of the subroutine;
- * Comments on subroutine use, preferably with an example, and one or more references for the algorithm;

- * List and definition of all local variables and subroutine arguments passed.

ROUTINE: Shell-Metzner Sort routine for sorting single arrays in ascending order.

REFERENCE: Miller, A. R., "Pascal Programs for Scientists and Engineers," 1982, Sybex, Berkeley, California.

COMMENTS: While the subroutines sort alphanumeric strings, it can be edited to sort numbers. Simply remove the occurrences of the dollar sign following the variable names.

PARAMETERS:

A\$() Array of strings to be sorted (IN/OUT). Assumed string length is 80 characters.

N Number of elements to be sorted must be less than or equal to the actual array limit (IN).

Listing of Shell-Metzner sort.

```

1010 K = N
1020 IF K <= 1 THEN 1130
1030 K = INT(K / 2)
1040 D = 1
1050 FOR J = 1 TO (N - K)
1060 I = J + K
1070 IF A$(J) <= A$(I) THEN 1100
1071 '
1080 H$ = A$(I)
1081 A$(I) = A$(J)
1082 A$(J) = H$
1090 D = 0
1100 NEXT J
1110 IF D = 0 THEN 1040
1120 GOTO 1020
1130 '
1140 RETURN

```

ROUTINE: Quicksort routine for sorting single arrays in ascending order.

REFERENCE: Miller, A. R., "Pascal Programs for Scientists and Engineers," 1982, Sybex, Berkeley, California.

COMMENTS: While the subroutines sort alphanumeric strings, it can be edited to sort numbers. Simply remove the occurrences of the dollar sign following the variable names.

PARAMETERS:

A\$() Array of strings to be sorted (IN/OUT). Assumed string length is 80 characters.

N Number of elements to be sorted must be less than or equal to the actual array limit (IN).

Listing of Quicksort.

```
1010 DIM L(20),R(20)
1020 L(1) = 1
      :R(1) = N
      :S = 1
1030 '
1031 IF S <= 0 THEN GOTO 1230
1040 IF L(S) >= R(S) THEN S=S + 1
      :GOTO 1220
1050 I = L(S)
      :J = R(S)
      :P$ = A$(J)
      :M = INT(I + J) / 2
1060 IF (J-I) > 5 THEN 1070 ELSE
1090
1070 IF ((A$(M) < P$) AND (A$(M) >
      A$(I)) OR ((A$(M) > P$) AND
      (A$(M) < A$(I))) THEN H$ =
      A$(M)
      :A$(M) = A$(J)
      :A$(J) = H$
1080 GOTO 1100
1090 IF ((A$(I) < A$( )) AND (A$(I)
      > P$) OR ((A$(I) > A$(M)) AND
      (A$(I) < P$)) THEN H$ = A$(I)
      :A$(I) = A$(J)
      :A$(J) = H$
1100 P$ = A$(J)
1110 '
1111 IF I >= J THEN 1170
1120 IF A$(I) < P$ THEN I = I + 1
      :GOTO 1120
```

```
1130 J = J - 1
1140 IF (I < J) AND (P$ < X(J))
      THEN 1130
1150 IF I < J THEN H$ = A$(I)
      :A$(I) = A$(J)
      :A$(J) = H$
1160 GOTO 1110
1170 J = R(S)
      :H$ = A$(I)
      :A$(I) = A$(J)
      :A$(J) = H$
1180 IF (I - L(S)) >= (R(S) - I)
      THEN 1190 ELSE 1200
1190 L(S+1) = L(S)
      :R(S+1) = I - 1
      :L(S) = I + 1
1191 GOTO 1210
1200 L(S+1) = I + 1
      :R(S+1) = R(S)
      :R(S) = I - 1
1210 S = S + 1
1220 '
1221 GOTO 1030
1230 '
1240 RETURN
```

A MATTER OF TIMING

These two short programs, submitted by Jay Holovacs of Warren, New Jersey, are ideal for that breed of programmer who 'always searching for the most efficient algorithm. -Ed.

LINCNT and LINLOG are a pair of utilities that analyze the performance of a BASIC program through summing and graphically describing the total length of time that the program spends executing each line. Normally, these routines are used after the BASIC program is already running well and ready for the final cleanup stage.

Preparation

LINCNT.BA installs the assembly language runtime component which does the actual timing. Run this routine once before starting your

program. Note that a call (CALL 60830) must be executed either as the first line of your program, or in the immediate mode shortly before each run. This resets and activates the runtime routine.

Run your program as usual. You should see no perceptible difference in the program's performance. As soon as possible afterwards, run LINLOG or POKE 62975,201 to remove the hook. The hook is normally passive in the immediate mode, however executing any additional numbered lines -- of any program -- will continue to add time to the counts until the hook is removed.

To conserve memory space and keep the routine simple, the timer only counts lines numbered between 0 and 1023, but this shouldn't be a serious problem. It is very unlikely that a 1000-line program could actually execute in the 16-32K RAM anyhow! Lines outside this range will simply be ignored.

Analyzing Results

Run LINLOG to examine the results. You will be prompted for an output file (normally a RAM file or LPT:) and descriptive information. Enter a version number, running circumstances for your own reference about the test. LINLOG's output consists of line numbers, time in seconds (+ indicates out of range -- over approx 254 seconds) and a histogram consisting of a graphic dashed line. If a time overflow occurred for any particular line, a ">" is placed at the end of the dashed line.

Further tests can be run without rerunning LINCNT. Be sure, though, to CALL 60830 before each execution to reset the routine.

Technical Notes

LINCNT uses a buffer of 2048 bytes to record activity of lines 0-1023. On every background

interrupt (approx 256 times per second) the routine checks to see if a line is being executed. If so, LINCNT increments the two-byte buffer location corresponding to that line number. The actual line number is actually identified only by offset from the array base.

Before running the test program, CALL 60830 activates the routine by clearing the 2048 bytes of buffer space and installing the interrupt hook into the background task.

For the curious, the source listing for the LINCNT.BA data is reproduced below:

Assembler listing for LINCNT.BA.

```
;INIT/LCHECK VER 1.1
;BY JAY HOLOVACS (CIS 74756,413)
;8/27/85
;CLEARS ARRAY OF 2048 BYTES
      ORG      .0ED9EH
LINE  EQU      63098      LINE # LOC
VECTOR EQU      0F5FFH    HOOK LOCATION
INIT   LXIH     ARRAY
      MVI      0          LOOP COUNTER
INIT1  MVIA     0          BYTE COUNTER
INIT2  MVIM     0
      INXH
      INRA
      JNZ      INIT2      INCR ADDRESS
      INRB
      MOVAB
      CPI      8          INCR BYTE
      JNZ      INIT1      NUMBER OF LOOPS
      LXIH     LCHECK
      SHLD     VECTOR+1    GET ADDRESS FIRST
      MVIA     195
      STA      VECTOR     JUMP INSTRUCTION
      RET
;*****
;
;ROUTINE TO GET LINE #
;CHECKS CURRENTLY EXECUTING LINE NUMBER
;AND INCREMENTS APPROPRIATE BYTE PAIR
;
LCHECK PUSHPSW
      PUSHB
      PUSHD
      PUSHH
      LHLD     LINE        GET LINE # INTO HL
      MVIA     03H         HIBYTE OF 3FFH (1023)
      CMPL
      JC       ENDRN       A IS SMALLER
      XCHG
      MOVAD
      RAL
      MOVD     DOUBLE HIBYTE
      MOVDA
      MOVAE
      RAL
      JNC      WRI         DOUBLE LOBYTE
      INRD
      CARRY INTO HIBYTE
```



```

WRI  MOVEA
      LXIH  ARRAY
      DADD  GET OFFSET FOR LINE #
      INRM  INCR LOBYTE
      JNZ   ENDRTN
      INXH
      INRM  INCR HIBYTE
      JNZ   ENDRTN NO OVERFLOW
      MVIM  0FFH USE 255 AS OVERFLOW FLAG
ENDRTN POPH
      POPD
      POPB
      POPPSW
      RET    COUNT COMPLETE
;
;
;VALUES STORED IN ARRAY, LO BYTE FIRST
ARRAY DEFB 0 BEGINNING OF BUFFER
;ACTUAL LENGTH 800H (2048) BYTES
END

```

Listing for LINCNT.BA, the hook installer.

```

1 REM LINCNT V1.1 8/27/85 BY JAY
  HOLOVACS (CIS74756,413)
2 REM Installs routine to time
  program execution reads only
  lines 0 to 1000
3 CLS
:PRINT "* LNCNT.BA V1.0 *"
:PRINT "This program occupies 2137
  bytes of"
:PRINT "high memory"
:PRINT
:PRINT "Place 'CALL 60830' at the"
:PRINT "start of program to be
  tested"
4 CLEAR 256,60829
5 DATA 33,235,237,6,0,62,0,54,0,
  35,60,194,165,237,4,120,254,8,
  194,163,237,33,191,237,34,0,246,
  62,195,50,255,245,201,245,197,
  213,229,42,122,246,62,3,188,218,
  230,237
6 DATA 235,122,23,87,123,23,210,
  214,237,20,95,33,235,237,25,52,
  194,230,237,35,52,194,230,237,54,
  255,225,209,193,241,201,0
7 FOR PP = 60830 TO 60907
:READ PX
:POKE PP,PX
:NEXT
8 PRINT
:PRINT "After running, run
  LINLOG.BA or POKE"
:PRINT "62975,201 to disable count"

```

Listing for LINLOG.BA, the LINCNT output program.

```

1025 REM LINLOG V1.1.2 9/14/85
  Use after running LINCNT in
  conjunction with a program
  (uses line #'S above those
  recognized by LINCNT)
1026 REM BY JAY HOLOVACS
  (CIS 74756,413)
1027 POKE 62975,201
:CLS
:PRINT "* * LINE # TIME LOGGING
  PROGRAM"
:PRINT " Use in conjunction
  with LINCNT.BA V1.1"
:PRINT
:INPUT "File for output data";F$
1028 DEFSNG A-Z
:OPEN F$ FOR OUTPUT AS 1
:LINE INPUT "Test program name
  and version:";PG$
:PRINT #1, "PROGRAM: ";PG$
:PRINT #1, "LINE      TIME (SEC)"
:PRINT #1,
  "-----"
:PRINT "Please wait.."
1029 FOR Q=60908 TO 62954 STEP 2
:H=PEEK(Q)
:IF H>HX THEN HX=H
1030 NEXT Q
:K=55/HX
1031 FOR Q=60907 TO 62954 STEP 2
:L=PEEK(Q)
:H=PEEK(Q+1)
:IF (H OR L)=0 THEN
1034 ELSE HL=H+L/256
:HT=HT+HL
1032 PRINT #1,(Q-60907)/2;CHR$(9);
:PRINT #1,USING "###.###";HL;
:PRINT #1,STRING$(K*H+1,45);
:IF H=255 THEN OF=1
:PRINT #1,">";
1033 PRINT #1,
1034 NEXT Q
:BEEP
:PRINT #1,
:PRINT #1,"TOTAL ACCUMULATED
  TIME=";
:PRINT #1,USING "###.###";HT;
:IF OF=1 THEN PRINT #1,"+";
1035 PRINT#1," SEC"
:CLOSE

```

A GREATER ROLE

The importance of personal computers to business and government planners can't be overstated. Microcomputers are slowly supplanting mainframes in Federal and State government, as well as in Big Business.

Now, even the U.S. economy is forecast by microcomputer. Chase Econometrics, a subsidiary of The Chase Manhattan Bank, uses a financial modeling program on an IBM PC/AT to make educated guesses about the macroeconomy.

Chase Econometric's PC-SIM software produces forecasts, models and graphics. Before Chase switched over to the PC/AT and PC-SIM, these reports were generated by a large multiuser Amdahl V-8 mainframe.

What next? Small-town fiscal planning with the Model 100?

Alan L. Zeichick
Editor

PORTABLE PROGRAM REVIEW

P.O. Box 250
Highland Mill
Camden, ME 04843

FIRST CLASS U.S. POSTAGE PAID CAMDEN, ME PERMIT NO. 5
--

SUBSCRIBER: PLEASE ADVISE IF ABOVE ADDRESS IS INCORRECT

PORTABLE PROGRAM REVIEW is published monthly and copyright (c) 1985, all rights reserved, by Camden Communications Inc., a Maine corporation with offices at Highland Mill (P.O. Box 250), Camden, Maine 04843, James S. Povec, President. Contents of this publication may not be reproduced in whole or in part unless expressly authorized in writing by the publisher. Tandy and Radio Shack are registered trademarks of Tandy Corporation. U.S. Subscription \$29.97. Please address all subscriptions and inquiries to Nancy Wight, Circulation Director, (207) 236-4365, CompuServe ID 76703,372. ISSN: 0883-7295.